Lecture 5: More on Decision Trees

- Measuring accuracy empirically
- Noise and overfitting
- Occam's razor, and why it makes sense
- Bells and whistles

Estimating the accuracy of a classifier

We want to estimate the *true error* of the classifier

I.e., the error it would make on *unseen data*

- **Resubstitution:** Build the classifier using all the training data, and test it using the same data

  Too optimistic! (why?)
- **Test sample estimation:** Divide the data into a *training set* and a *test set*

  Wastes data
- **Cross-validation:** General method for determining accuracy

k-fold cross-validation procedure

1. Split the training data into k partitions (folds), ensuring that the class distribution is roughly the same in each partition
2. Repeat k times:
   (a) Take one fold to be the test set
   (b) Take the remaining $k-1$ folds to form the training set
   (c) We train the decision tree on the training set, then measure $TrainingError_i$ and $TestError_i$
3. Report the average of $TrainingError_i$ **and** the average of $TestError_i$, $i = 1, \ldots k$.

Magic number: $k = 10$.

More about cross-validation

- *Leave-one-out cross-validation:* special case in which the test is performed on just one instance.
  Used especially if data is scarce.
- If for any reason we need a validation set (for the algorithm itself), that will be kept separate from the training and test sets
  E.g. One fold is for testing, one for validation and the remaining $k-2$ for training
- If we are comparing different algorithms *test them on the SAME folds!*

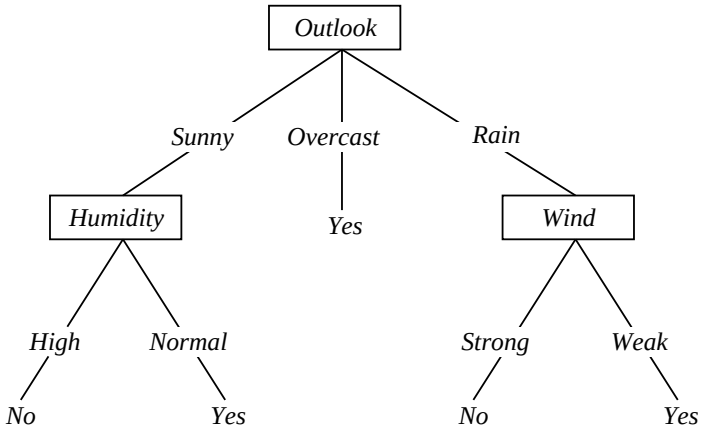
Dealing with noise in the training data

Noise is inevitable!

- Values of attributes can be misrecorded
- Values of attributes may be missing
- The class label can be misrecorded

What happens when adding a noisy example?

Example: The effect of noise



Suppose we add to the data a noisy example:

Sunny, Hot, Normal, Strong, PlayTennis=No

The tree grows unnecessarily!

Overfitting

Consider error of hypothesis h over

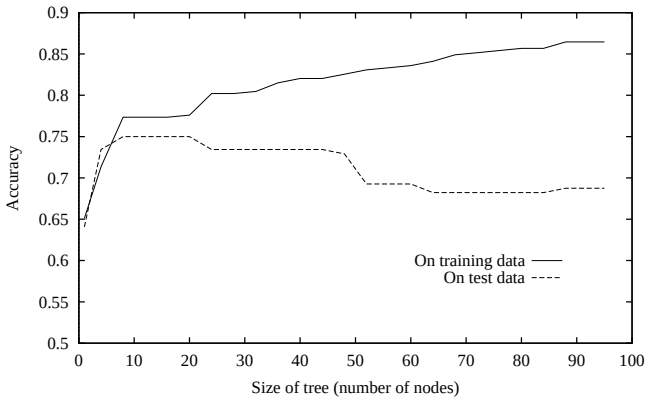
- Training data: $error_{train}(h)$
- Entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis h **overfits** training data if there is an alternative hypothesis h' such that

$$error_{train}(h) < error_{train}(h') \text{ and } error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

This is a general problem for all supervised learning methods!

Overfitting in decision trees



As the tree grows, the accuracy degrades, because the algorithm is finding *irrelevant* attributes.

Do not believe anyone's results unless they report them on separate training and test sets!

Avoiding overfitting

1. Stop growing the tree when further splitting the data does not yield a statistically significant improvement
2. Grow a full tree, then *prune* the tree, by eliminating nodes

The second approach has been more successful in practice

In both cases, the leaves of the tree will now be impure:

- The leaf can be assigned the class label of the majority of the instances which reached the leaf
- Alternatively, one can use probability estimates of the class membership, based on instance counts.

How to select the “best” tree

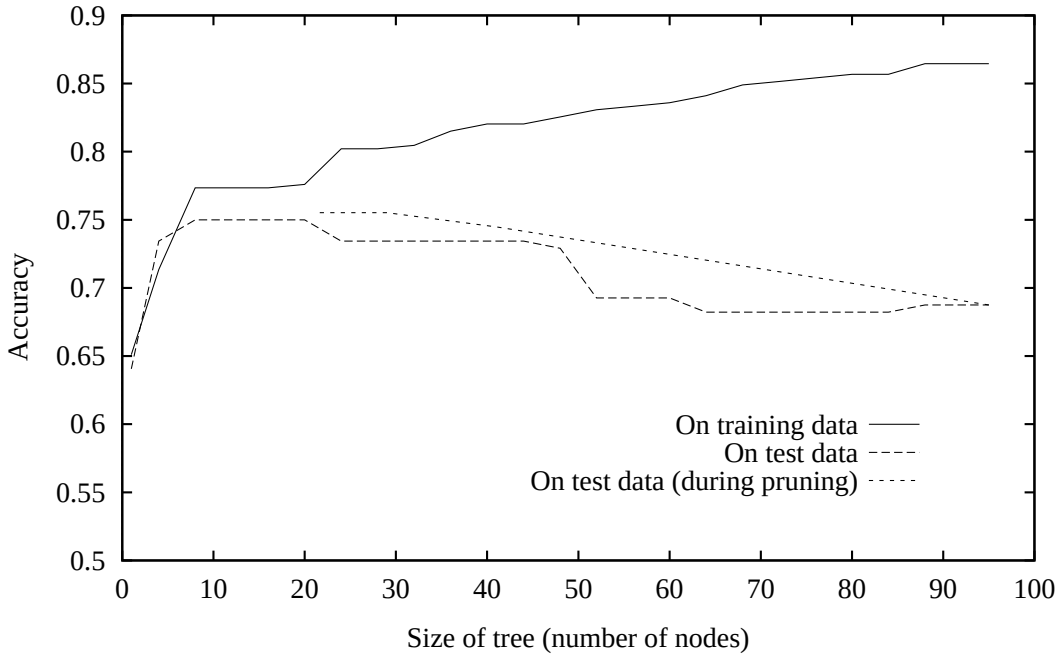
1. Measure performance over training data only
2. Measure performance over a separate validation data set
3. Minimum description length (MDL) principle

The second one (training and validation set) is the most common.

Example: Reduced-error pruning

1. Split the “training data” into a *training set* and a *validation set*
2. Grow a large tree (e.g. until each leaf is pure)
3. For each node:
   (a) Evaluate the validation set accuracy of pruning the subtree rooted at the node
   (b) Greedily remove the node that most improves validation set accuracy, with its corresponding subtree
   (c) Replace the removed node by a leaf with the majority class of the corresponding examples.
4. Stop when pruning starts hurting the accuracy on the validation set.

Example: Effect of reduced-error pruning



Example: Rule post-pruning in C4.5

1. Convert the decision tree to rules
2. Prune each rule independently of the others, by removing preconditions such that the accuracy is improved
3. Sort final rules in order of estimated accuracy

C4.5 builds a pessimistic estimate based on the accuracy on the training set.

Advantages:

- Can prune attributes higher up in the tree *differently on different paths*
- There is no need to reorganize the tree if pruning an attribute that is higher up
- Most of the time people want rules anyway, for readability

Minimum Description Length (MDL) Principle

William of Ockham (1285-1347): Entities are not to be multiplied beyond necessity (law of parsimony).

Application to machine learning (and science in general)

Prefer simpler hypotheses over more complicated ones

People like to look at small hypotheses, but does this really make sense?

Revisiting the friendly Bayesian framework...

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(D|h)P(h) \\ &= \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \end{aligned}$$

We know from information theory that the optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.

So we can interpret our equation:

- $-\log_2 P(h)$ is length of h under the optimal code
- $-\log_2 P(D|h)$ is length of D given h under the optimal code

Minimum Description Length Principle (MDL)

$$h_{MAP} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

MDL : suppose I want to send my data D over. The best hypothesis is the one that minimizes the length of the message to send.

- We will send the hypothesis (encoded optimally)
- If data is correctly labeled by the hypothesis, we do not need to send it anymore (h summarizes it)
- But we do need to send the misclassified data

But this is precisely the MAP hypothesis!

MDL applied to decision trees

$$h_{MDL} = \arg\min L(\text{tree}) + L(\text{examples misclassified})$$

The MDL principle trades off tree size for training errors.

If a tree is small, it can make some training errors, and the message produced would still be more compact.

Missing values during classification

- Assign “most likely” value based on all the data that reaches the current node. “Most likely” means the most frequent attribute value
- Assign all possible values with some probability. Usually we just count the occurrences of the different attribute values in the instances that have reached the same node. We will predict all the possible class labels with the appropriate probabilities too.

Missing values during tree construction

1. Introduce an “unknown” value
2. Modify gain ratio to take into account the probability of an attribute being known:

$$Gain(S, A)P(A)$$

where P(A) is the fraction of the instances that reached the node, in which the value was known

Costs of attributes

Include cost in the metric, e.g.

$$\frac{Gain^2(S, A)}{Cost(A)}$$

Mostly a problem in specific domains (e.g. medicine).

Multiple metrics have been studied and proposed, without a consensus.